Mastering Symfony

SensioLabs
Créateur de Symfony

Training program

I. Services

II. Console

III. Security

IV. Events

Services

SensioLabs
Créateur de Symfony

Training program

I. Introduction

II. Retrieve services

III. Dependency injection

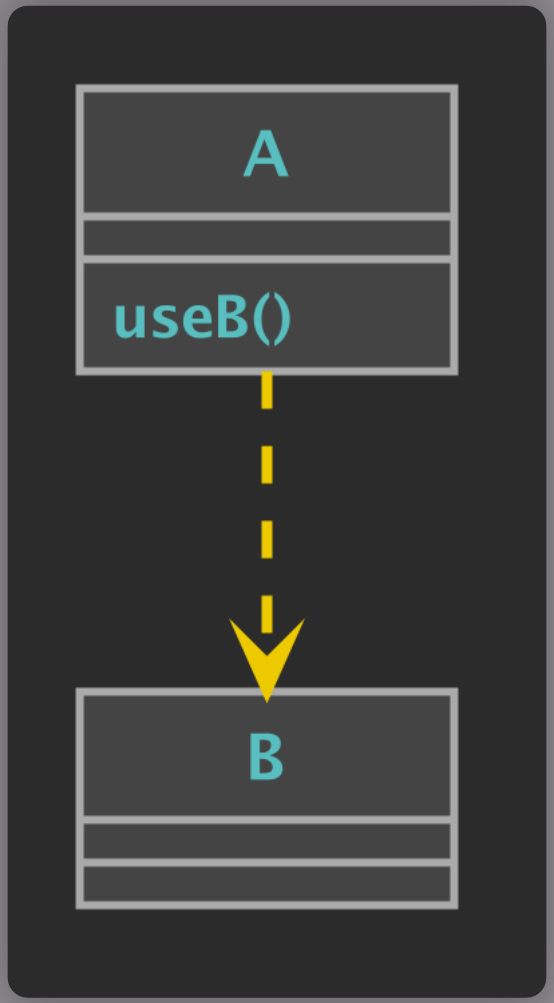
IV. Service configuration

Introduction

What are we talking about?

- Think about a logger…
- Do you configure it by yourself?
- What about a custom logger?
- How to provide it to a controller?

Dependency



About services

- Services are:
  - Configuration
  - Related to PHP classes
  - Handled by the Symfony DependencyInjection component
  - Descriptions of how to instantiate PHP classes
  - Most of the time already dynamically scanned by Symfony: no manual conf

Have a look

```php
/*
 * Assuming:
 *   - BookManager IS a custom service,
 *   - EntityManagerInterface IS a vendor service,
 * there is no need any configuration by default in Symfony
 * to provide the argument of the constructor.
 *
 * THIS is called autowiring.
 */
class BookManager
{
    /** @var \Doctrine\ORM\EntityManagerInterface */
    private $entityManager;

    public function __construct(EntityManagerInterface $entityManager)
    {
        $this->entityManager = $entityManager;
    }
}
```

Retrieve services

Useful commands

```
symfony console debug:autowiring --all
symfony console debug:autowiring mail

symfony console debug:container Symfony\Component\Mailer\MailerInterface
symfony console debug:container mailer.mailer
symfony console debug:container mailer.mailer --show-arguments
```

Dependency injection

How to get services

- Container:
  - Dynamic way to retrieve exposed services
  - $this->container->get('service_id')
- Autowiring:
  - Static and optimized way to inject services

Store dependencies in a service

- Constructor injection
- Setter injection
- Property injection

Autowiring in a constructor

```php
/*
 * Assuming:
 *   - BookManager IS a custom service,
 *   - EntityManagerInterface IS a vendor service,
 * there is no need any configuration by default in Symfony
 * to provide the argument of the constructor.
 *
 * THIS is called autowiring.
 */
class BookManager
{
    /** @var \Doctrine\ORM\EntityManagerInterface */
    private $entityManager;

    public function __construct(EntityManagerInterface $entityManager)
    {
        $this->entityManager = $entityManager;
    }
}
```

Autowiring in a setter

```php
class BookManager
{
    /** @var \Doctrine\ORM\EntityManagerInterface */
    private $entityManager;

    /**
     * @required
     */
    public function setEntityManager(EntityManagerInterface $entityManager): void
    {
        $this->entityManager = $entityManager;
    }
}
```

Autowiring in a property

```yaml
# config/services.yaml

services:
    App\BookManager:
        properties:
            manager: '@Doctrine\ORM\EntityManagerInterface'
```

```php
class BookManager
{
    /** @var \Doctrine\ORM\EntityManagerInterface  */
    private $entityManager;
}
```

Tips

- Rule 1: Use constructor injection whenever possible.
- Rule 2: Use setter injection for specific cases.
- Rule 3: Use property injection as a last resort.

Only for information

```php
<?php

// var/cache/dev/ContainerNpez2qb/getBookManagerService.php

use Symfony\Component\DependencyInjection\Argument\RewindableGenerator;

// This file has been auto-generated by the Symfony Dependency Injection Component for internal use.
// Returns the public 'App\BookManager' shared autowired service.

include_once $this->targetDirs[3].'/src/BookManager.php';

return $this->services['App\\BookManager'] = new \App\BookManager(
  ${($_ = isset($this->services['doctrine.orm.default_entity_manager']) ?
     $this->services['doctrine.orm.default_entity_manager'] :
     $this->load('getDoctrine_Orm_DefaultEntityManagerService.php')) && false ?: '_'}
);

```

Binding arguments

```yaml
services:
    _defaults:

        # additional context provided to autowired services
        bind:
            int $someValue: 10
```

Service configuration

Default configuration

```yaml
services:
    _defaults:
        autowire: true      # Automatically injects dependencies in your services.
        autoconfigure: true # Automatically registers your services as commands & event subscribers

    # Makes classes in src/ available to be used as services
    # this creates a service per class whose id is the fully-qualified class name
    App\:
        resource: '../src/*'
        exclude: '../src/{DependencyInjection,Entity,Migrations,Tests,Kernel.php}'

    # Controllers are imported separately to make sure services can be injected
    # as action arguments even if you don't extend any base controller class
    App\Controller\:
        resource: '../src/Controller'
        tags: ['controller.service_arguments']
```

Manual configuration

```yaml
services:
    app.book_manager:
        class: App\BookManager
        arguments:
            - '@doctrine.orm.default_entity_manager'
        public: false

    App\BookManager:
        alias: 'app.book_manager'
        public: false
```

Manual configuration

- But…

Best-practice

Use Autowiring to automate the configuration of application services

Environment variables

```
$ symfony console debug:container --env-var=SOME_KEY

Symfony Container Environment Variables
=======================================

 // Displaying detailed environment variable usage matching SOME_KEY

 None of the environment variables match this name.
```

Environment variables

```
# .env file (or any override such as .env.local)
# Set an environment variable if missing in the host

SOME_KEY='37fd4a3c83ab2'
```

```
# app/services.yaml

parameters:
    # Fallback if missing in your .env files
    env(SOME_KEY): 'aaaaaaaaaaaa'

services:
    App\Demo:
        arguments:
            $key: '%env(string:SOME_KEY)%'
```

Environment variables

```
$ symfony console debug:container --env-var=SOME_KEY

Symfony Container Environment Variables
=======================================

 // Displaying detailed environment variable usage matching SOME_KEY

%env(string:SOME_KEY)%
----------------------

 ----------------- ----------
  Default value     "aaaaaaaaaaaa"
  Real value        "37fd4a3c83ab2"
  Processed value   "37fd4a3c83ab2"
 ----------------- ----------

 // Note real values might be different between web and CLI.
```

Exercises

1. Retrieve and configure our OMDb API consumer.
2. Create a new service to fetch movie data from the API.

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/service_container.html

https://symfony.com/doc/current/configuration/env_var_processors.html

Console

SensioLabs
Créateur de Symfony

Training program

I. Context

II. Custom commands

III. Helpers

Context

Context

- Many use cases:
  - Create some code snippets
  - Get stats about the project
  - I/O on the database
  - Sync data
  - . . .

Features

- Modes:
  - Interactive
  - Batch
- Support:
  - Debug stages
  - Kernel and app access
  - Specific events
  - Helpers
  - Test kit
- Remember: The Command class IS a service

Usage

```
$ symfony console help

Description:
  Displays help for a command

Usage:
  help [options] [--] <command> [<command_name>]

Arguments:
  command               The command to execute
  command_name          The command name [default: "help"]

Options:
      --format=FORMAT   The output format (txt, xml, json, or md) [default: "txt"]
      --raw             To output raw command help
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
  -e, --env=ENV         The Environment name. [default: "dev"]
      --no-debug        Switches off debug mode.
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output, 2 for more verbose
output and 3 for debug

```

Custom commands

New empty command

```php
<?php

namespace App\Command;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

class BookFindCommand extends Command
{

    protected function configure(): void
    {
        $this
            ->setName('app:book:find')
        ;
    }

    protected function execute(InputInterface $input, OutputInterface $output): int
    {
        return 0;
    }
}

```

Use the command

```
1 $ symfony console app:book:find
```

Configuration

```
1     protected function configure(): void
2     {
3         $this
4             ->setName('app:book:find')
5             ->setAliases(['book:find'])
6             ->setDescription('Find a book by isbn')
7             ->setHelp('...')
8         ;
9     }
```

Options and arguments

- Each command can have:
  - Arguments
  - Options:
    - Long options: --help
    - Short options: --h
    - Combined short options: -abc (equivalent to: -a -b -c)
- Values can be given:
  - When calling the command
  - Interactively during the command process

Arguments

```php
protected function configure(): void
{
    $this
        ->setName('app:book:find')
        ->addArgument('isbn', InputArgument::REQUIRED, 'ISBN code required to retrieve a book')
    ;
}

protected function execute(InputInterface $input, OutputInterface $output): int
{
    $output->writeln('ISBN: ' . $input->getArgument('isbn'));

    return 0;
}
```

Arguments: usage

```
$ symfony console app:book:find

  Not enough arguments (missing: "isbn").

app:book:find [-h|--help] [-q|--quiet] [-v|vv|vvv|--verbose] [-V|--version] [--ansi] [--no-ansi] [-n|--no-interaction] [-e|--env ENV] [--no-debug] [--] <command> <isbn>

exit status 1


$ symfony console app:book:find "2-266-11151-5"
ISBN: 2-266-11151-5
```

Modes

- Argument modes:
  - InputArgument::REQUIRED
  - InputArgument::OPTIONAL
  - InputArgument::IS_ARRAY
- Option modes:
  - InputOption::VALUE_REQUIRED
  - InputOption::VALUE_OPTIONAL
  - InputOption::VALUE_NONE
  - InputOption::VALUE_IS_ARRAY

Helpers

About helpers

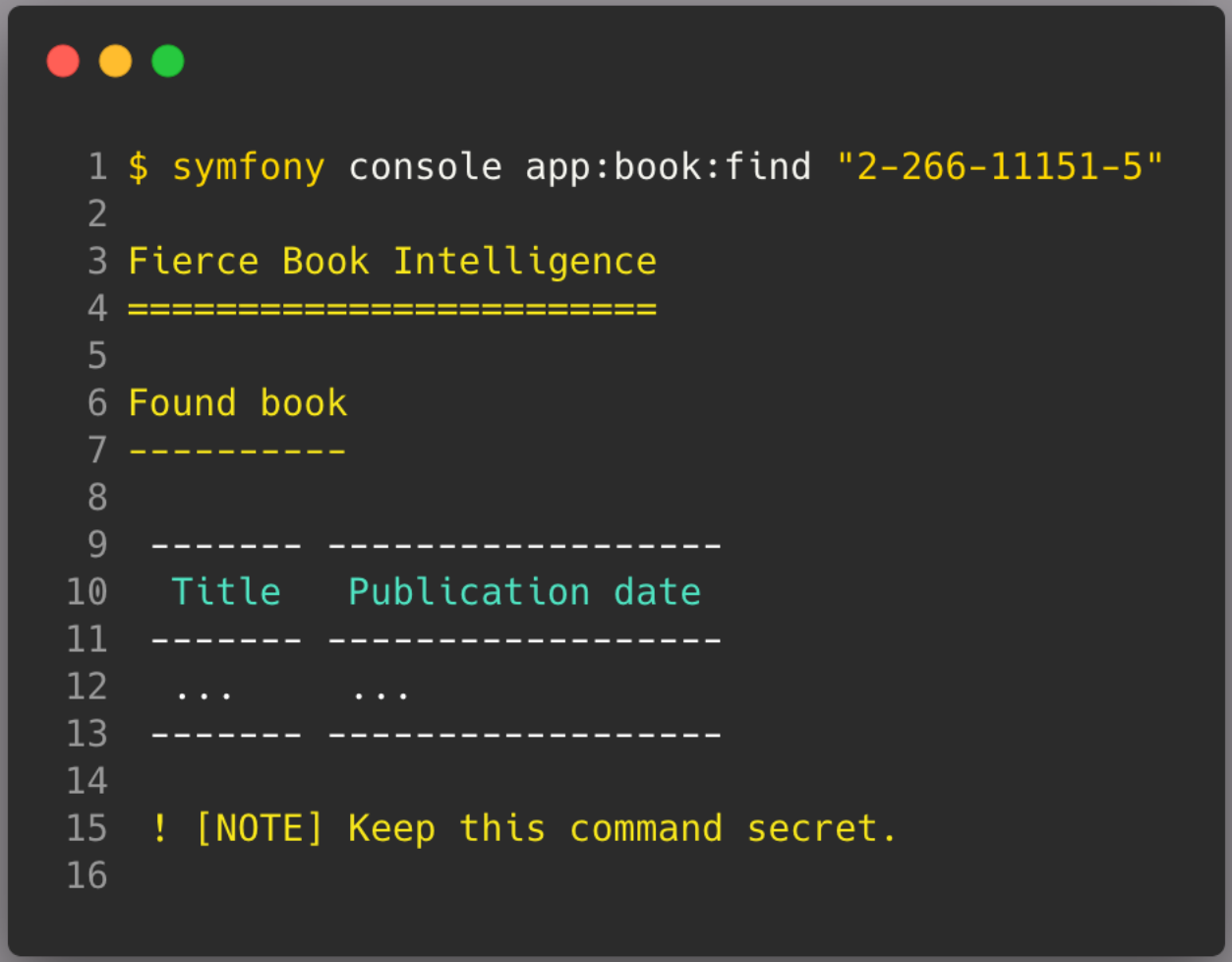
- Many helpers
- One meta-helper: SymfonyStyle

SymfonyStyle

```php
protected function execute(InputInterface $input, OutputInterface $output): int
{
    $io = new SymfonyStyle($input, $output);
    $io->title('Fierce Book Intelligence');

    $io->section('Found book');
    $io->table([
        'Title',
        'Publication date',
    ], [
        ['...', '...']
    ]);

    $io->note('Keep this command secret.');

    return 0;
}
```

See more: https://symfony.com/doc/current/console/style.html

SymfonyStyle - Output



```
$ symfony console app:book:find "2-266-11151-5"

Fierce Book Intelligence
========================

Found book
----------

 ------- ------------------
  Title   Publication date
 ------- ------------------
  ...     ...
 ------- ------------------

 ! [NOTE] Keep this command secret.

```

Fine tuned helpers

- Many helpers:
  - Question helper
  - Table
  - Progress bar
  - Process helper
  - Formatter helper
  - Debug formatter helper

See more: https://symfony.com/doc/current/components/console/helpers/index.html

Exercises

1. Create a new command to perform a data import for OMDB API into our database.
    1. The command will ask for either a OMBD id or a movie title
    2. The command will display the steps processed, and the movie information once imported, such as:
        1. ids (the one from the API and ours from the database)
        2. title
        3. MPAA restriction (Movie::rated)

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/console.html

Security

SensioLabs
Créateur de Symfony

Training program

I. Introduction

II. Authentication and Guard

III. Native authorization with roles

IV. Custom voters

Introduction

Keywords

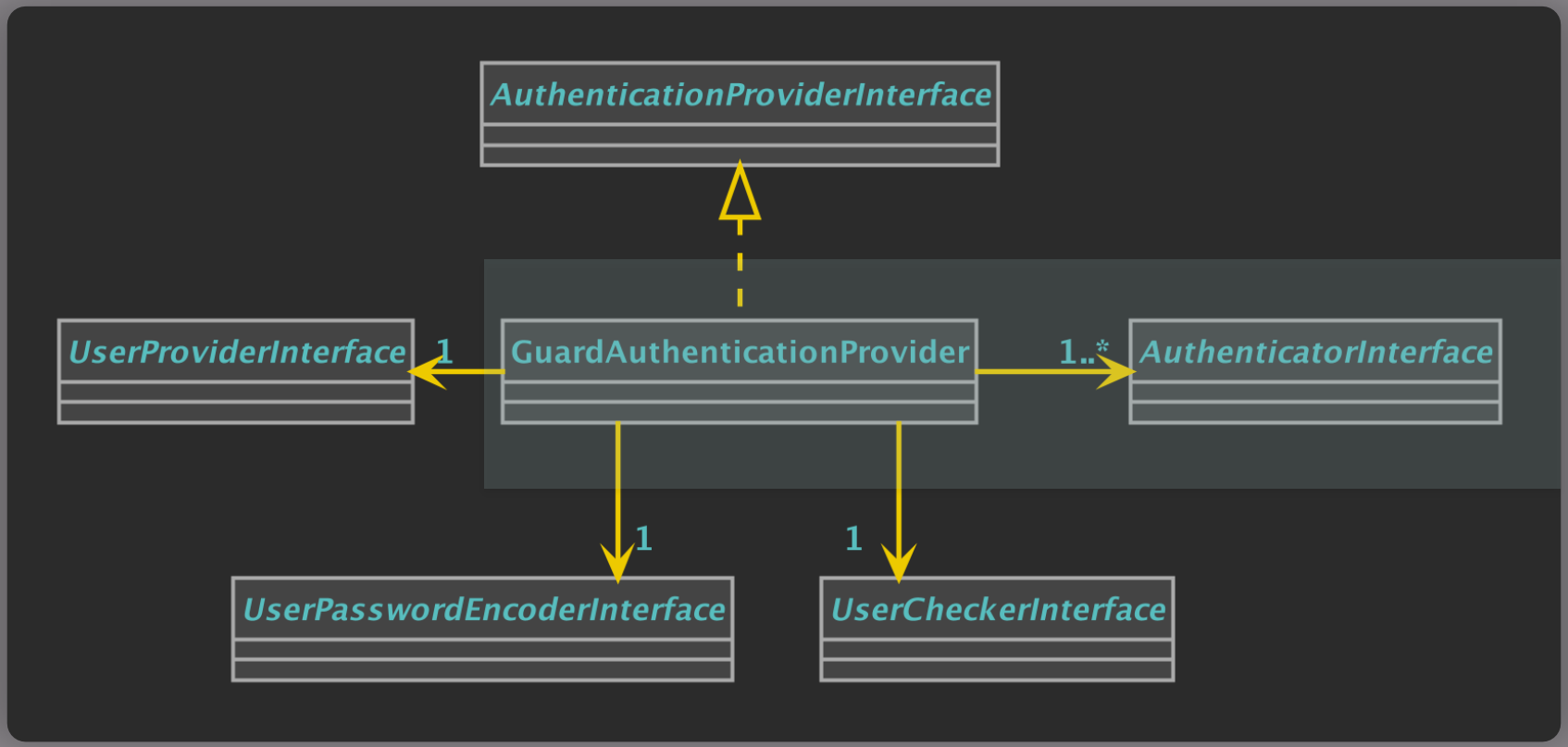
- 2 layers:
  - Authentication: Who is the user?
  - Authorization: According to the user's profile, does he pass some conditions?
- Authentication:
  - User provider: Users stored in the system
  - Authentication providers: Way to retrieve user profiles into tokens
- Authorization:
  - AccessDecisionManager: Referee of the voting system
  - Voters: Services that answer (yes, no, no idea) to a given question
- Firewall
- Helpers (See later)

Authentication and Guard

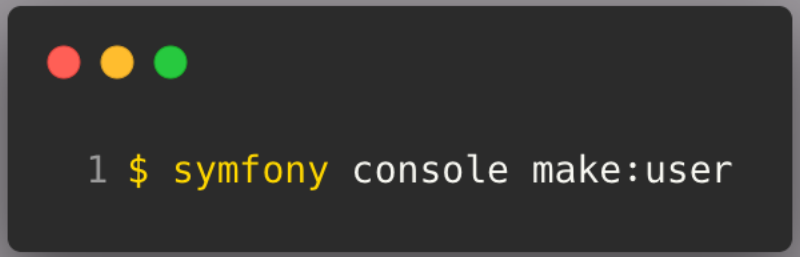
Authentication

- Authentication providers:
  - Built-in:
    - Form login (LDAP or not)
    - Json login
    - HTTP Basic
    - . . .
  - Custom:
    - GuardAuthenticationProvider: Allows you to have custom authenticators

Guard


AuthenticationProviderInterface
UserProviderInterface
1
GuardAuthenticationProvider
1..*
AuthenticatorInterface
1
1
UserPasswordEncoderInterface
UserCheckerInterface

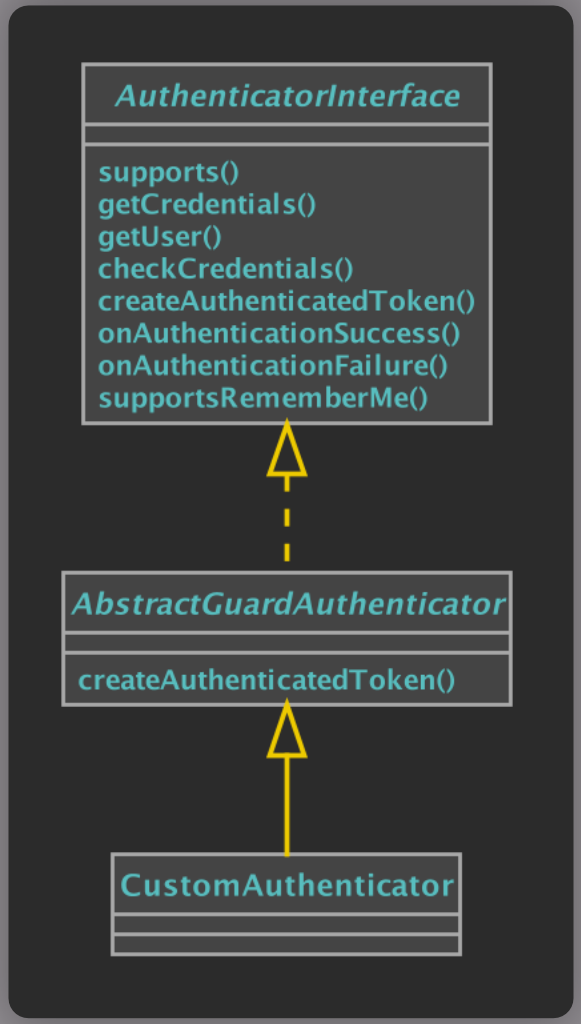
User model



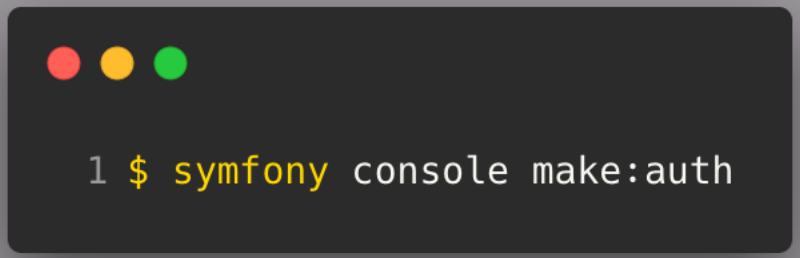
User provider

```yaml
# config/packages/security.yaml

# make:user updated this file (assuming we chose Doctrine as storage)
# Have a look on src/Entity/User.php as well 🤷

security:
    encoders:
        App\Entity\User:
            algorithm: auto

    # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
```

Authenticators



Custom authenticator



Configuration

```yaml
# config/packages/security.yaml

# On the "main" firewall,
#  make:user specified which user provider to choose
#  make:auth added our custom authenticator

firewalls:
    main:
        anonymous: lazy
        provider: app_user_provider
        guard:
            authenticators:
                - App\Security\MainAuthenticator
        logout:
            path: app_logout
```

Update your custom authenticator

- Open and edit your fresh authenticator
  - Located in src/Security

Native authorization with roles

How to check authorization?

- For each matching request:
  - See access_control in your security.yaml file
- Before calling a controller:
  - See @IsGranted() annotation
- In a controller:
  - See $this->isGranted(), or $this->denyUnlessGranted()
- In a service:
  - See AuthorizationChecker::isGranted(), or Security::isGranted()
- In a template:
  - See is_granted() function

What to check?

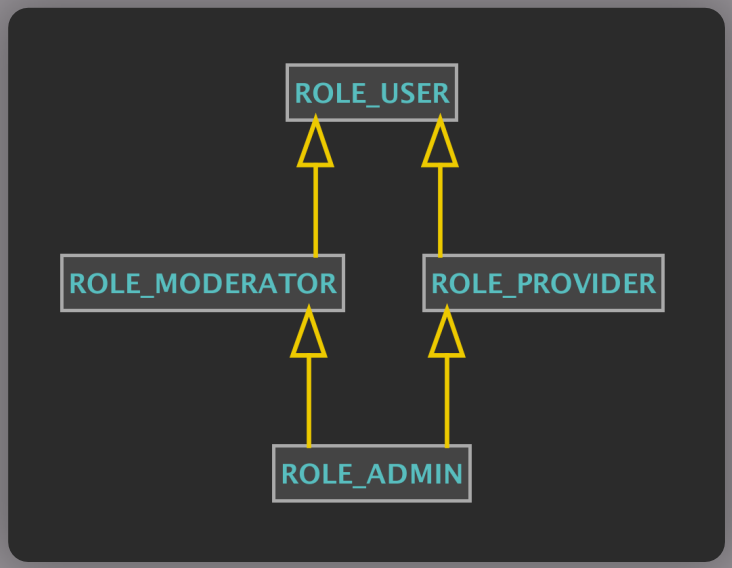
- Roles, by default: ROLE_XXXXXX
  - Role names are totally free, but are prefixed by ROLE_
- Operations on a given subject: (“Custom voters” chapter)
  - For example: { publish ; $book }

How to answer?

- Thanks to voters
- Built-in voters such as:
  - RoleVoter
  - RoleHierarchyVoter
- Custom voters

Roles - Hierarchy

```yaml
# config/packages/security.yaml

security:
    role_hierarchy:
        ROLE_USER: ~
        ROLE_MODERATOR: ROLE_USER
        ROLE_PROVIDER: ROLE_USER
        ROLE_ADMIN:
            - ROLE_MODERATOR
            - ROLE_PROVIDER

```



Roles - Storage

- UserInterface::getRoles(): array

Access control

```yaml
# config/packages/security.yaml

security:
    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
        - { path: ^/profile, roles: ROLE_USER }

```

See more: https://symfony.com/doc/current/security/access_control.html

Controller and authorization

```php
// . . .
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;

class BookController extends AbstractController
{
    /**
     * @Route("/book/{book}", name="book_update", methods={"GET"})
     * @IsGranted("ROLE_USER")
     */
    public function update(Request $request, Book $book): Response
    {
        dump($this->isGranted('ROLE_USER'));

        $this->denyAccessUnlessGranted('ROLE_USER');

        // . . .
    }
}

```

Service and authorization

```php
class BookService
{
    private $authorizationChecker;

    public function __construct(AuthorizationCheckerInterface $authorizationChecker)
    {
        $this->authorizationChecker = $authorizationChecker;
    }

    public function doSomeAdminTask(): void
    {
        if (!$this->authorizationChecker->isGranted('ROLE_ADMIN') {
            // . . .
        }
    }
}

```

Template and authorization

```twig
{% if is_granted('ROLE_USER') %}
    <div>Welcome, {{ app.user.firstname }}</div>
{% endif %}
```

Custom voters

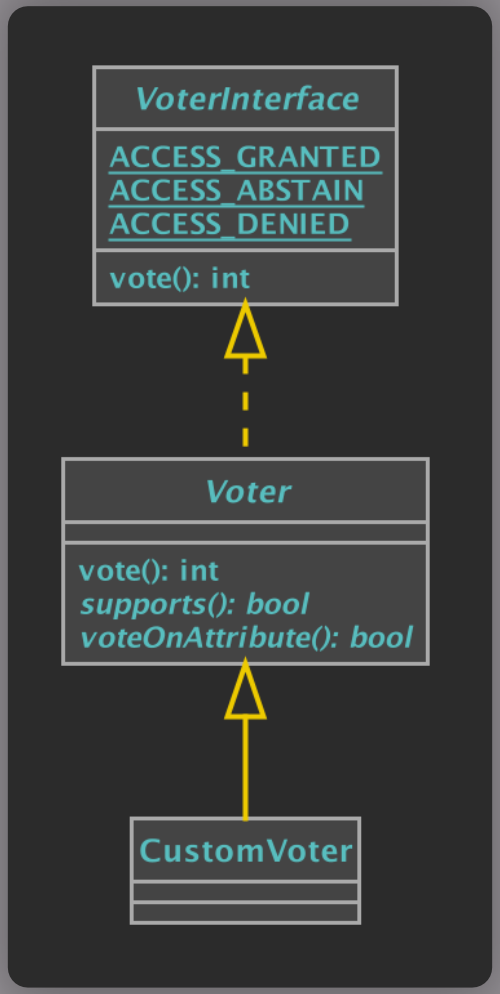
Context

```php
class BookController extends AbstractController
{
    /**
     * @Route("/book/{book}", name="book_update", methods={"GET"})
     */
    public function update(Request $request, Book $book): Response
    {
        $this->denyAccessUnlessGranted('update', $book);

        // . . .
    }
}

```

How to answer?

- By default, no voters can answer yes or no to the question with:
  - Attributes: ‘update’
  - Subject: $book
- By default, the app will finally answer “no”
- Let’s create a voter, then!

Voter


VoterInterface
ACCESS_GRANTED
ACCESS_ABSTAIN
ACCESS_DENIED
vote(): int
Voter
vote(): int
supports(): bool
voteOnAttribute(): bool
CustomVoter

New voter

```
1 $ symfony console make:voter
```

See more: https://symfony.com/doc/current/security/voters.html

Exercises

1. Create a custom Guard authenticator based upon a User managed by Doctrine.
2. Filter users so that only administrators can access ^/admin URLs.
3. Ensure a customer cannot order a movie if he doesn't meet age requirements.

The MPAA defines the following movie restrictions:
- "PG", "PG-13" require 13 y.o. to watch the movie
- "R", "NC-17" require 17 y.o. to watch the movie

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/security.html

https://symfony.com/doc/current/components/security/authentication.html

https://symfony.com/doc/current/security/auth_providers.html

https://symfony.com/doc/current/security/guard_authentication.html

https://symfony.com/doc/current/security/voters.html

Events

SensioLabs
Créateur de Symfony

Training program

I. Introduction

II. Subscribers

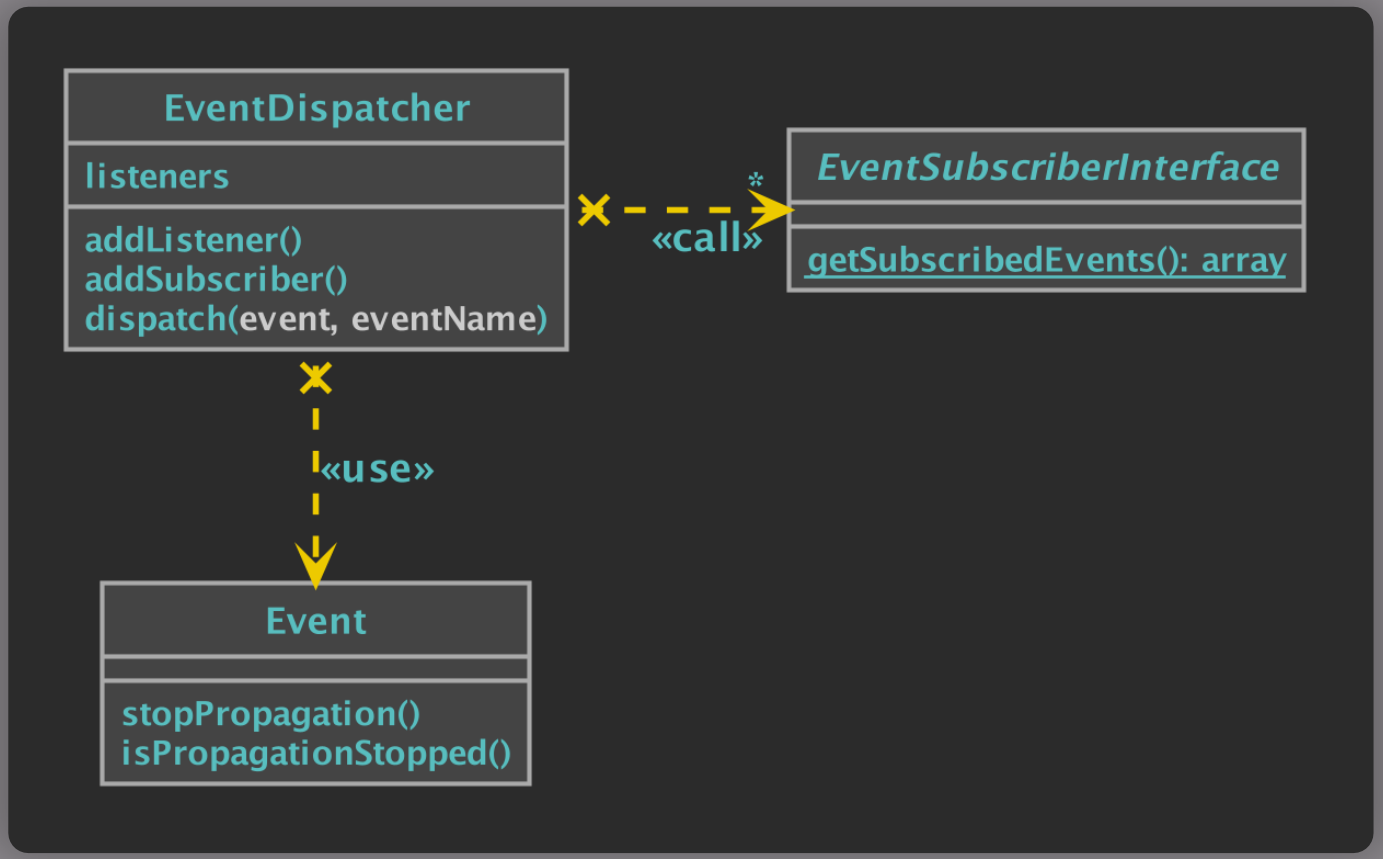
III. Events

Introduction

What are we talking about?

- Pluggable behavior of an algorithm
- Triggered events on observed actions:
  - Retrieve navigation data before rendering a menu
  - Automatically look for the user locale
  - . . .

Overview


EventDispatcher
listeners
addListener()
addSubscriber()
dispatch(event, eventName)
«call»
*
EventSubscriberInterface
getSubscribedEvents(): array
«use»
Event
stopPropagation()
isPropagationStopped()

Listeners

```
$ symfony console debug:event-dispatcher

Registered Listeners Grouped by Event
=====================================

"Symfony\Component\Mailer\Event\MessageEvent" event
---------------------------------------------------

 ------- ------------------------------------------------------------------------ ----------
  Order   Callable                                                                 Priority
 ------- ------------------------------------------------------------------------ ----------
  #1      Symfony\Component\Mailer\EventListener\MessageListener::onMessage()      0
  #2      Symfony\Component\Mailer\EventListener\EnvelopeListener::onMessage()     -255
  #3      Symfony\Component\Mailer\EventListener\MessageLoggerListener::onMessage()  -255
 ------- ------------------------------------------------------------------------ ----------

"Symfony\Component\Security\Http\Event\LogoutEvent" event
---------------------------------------------------------

. . .

```

Event subscribers

Subscribers

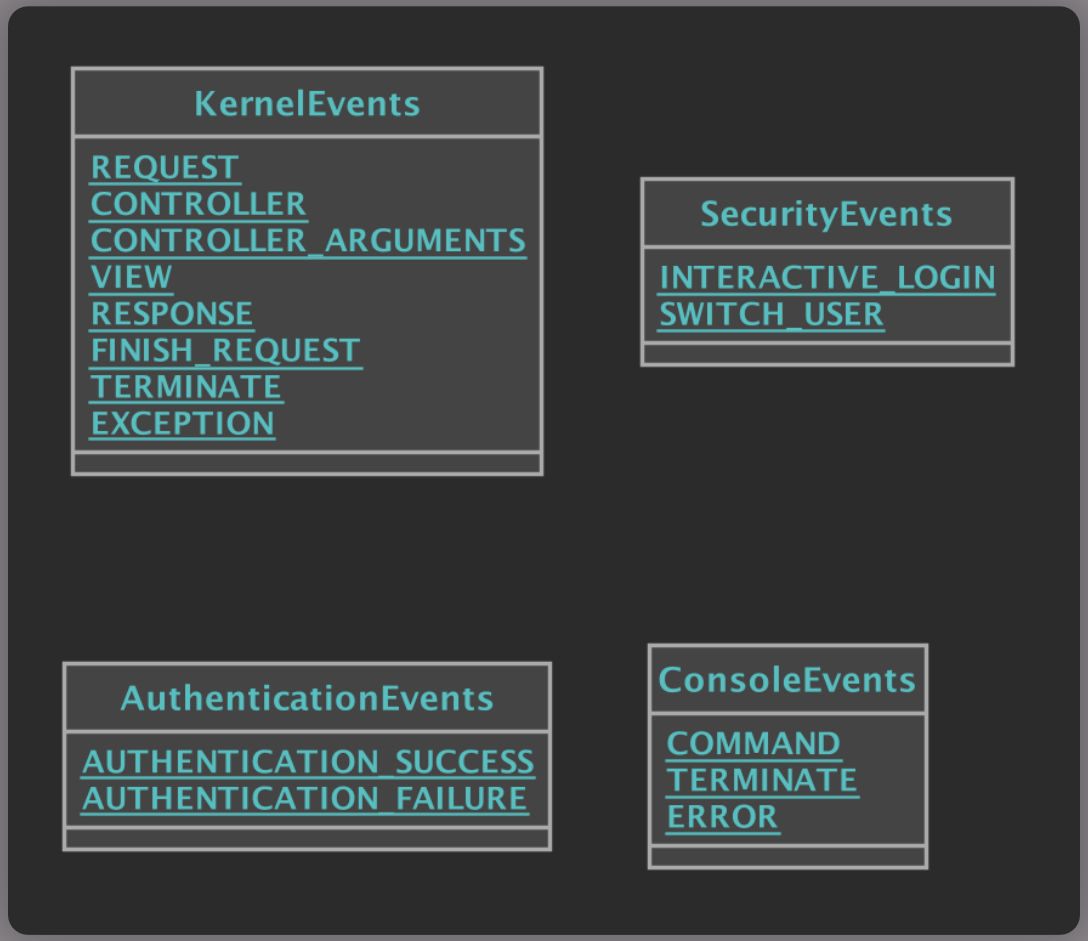
- Way to register several listeners at once
- Requires 0 configuration

Subscribers

```php
<?php

namespace App\Book\Event;

use Acme\Store\Event\BookEvent;
use Symfony\Component\EventDispatcher\EventSubscriberInterface;
use Symfony\Component\HttpKernel\Event\ResponseEvent;
use Symfony\Component\HttpKernel\KernelEvents;

class BookSubscriber implements EventSubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return [
            KernelEvents::RESPONSE => [
                ['onKernelResponsePre', 10],
                ['onKernelResponsePost', -10],
            ],
            BookEvents::PUBLISHED => 'onBookPublished',
        ];
    }

    public function onKernelResponsePre(ResponseEvent $event) {}
    public function onKernelResponsePost(ResponseEvent $event) {}
    public function onBookPublished(BookEvent $event) {}
}

```

Events

Symfony events



Main Symfony events explained

- Kernel events:
  - REQUEST: Filters the HTTP request and allows to return an early response.
  - CONTROLLER: Guesses the controller to call afterwards.
  - RESPONSE: Allows any updates in the HTTP response.
  - TERMINATE: Last tasks remaining after the response has been sent to the user.

- Security events:
  - INTERACTIVE_LOGIN: Triggered when the user manages to authenticate.
  - AUTHENTICATION_SUCCESS
  - AUTHENTICATION_FAILURE
  - SWITCH_USER: Triggered when a user switches to someone else's account.

See more: https://symfony.com/doc/current/components/http_kernel.html

Exercises

1. Store into your database the last connection time of your users.
   - Try to store the last time a user *actually* logs in.
2. Create a new log entry (in the default log file) each time a user orders a movie, thanks to an event listener.

SensioLabs

Créateur de Symfony

Resources

https://symfony.com/doc/current/components/event_dispatcher.html

https://symfony.com/doc/current/components/http_kernel.html

About us

We are the **creators of Symfony**. We know the framework and PHP inside out and we can help you.

Contact us

SensioLabs France
92-98, Boulevard Victor Hugo
92 115 Clichy Cedex
France
Tel: +33 (0)1 40 99 81 09
contact@sensiolabs.com

SensioLabs Deutschland
Neusser Str. 27-29
50670 Köln
Germany
Tel: +49 221 66 99 27 50
contact@sensiolabs.de

We also provide worldwide **on-site services**. Contact us at: sensiolabs.com/en/contact

SensioLabs Consulting

- We develop **proof of concept applications** to evaluate Symfony for your product.
- We **coach your team** and accompany your company through the development.
- We deliver **expert missions** to help you solve specific problems in your development.
- We help you **migrate** your **legacy** PHP applications.

Our full list of services: sensiolabs.com/en/packaged-solutions

SensioLabs Training

- We provide **general Symfony training** for all levels (Getting Started, Mastering and Hacking) and **specific Symfony training** for testing, performance and internals.
- We provide special Symfony training for your **developers**, including web development and Twig integration.
- We provide training for other **PHP** technologies such as Doctrine and Drupal.

Our full list of courses: training.sensiolabs.com

SensioLabs Training Department

Address

92-98 Boulevard Victor Hugo

92 115 Clichy Cedex

France

Phone

+33 1 40 99 82 05

Email

training@sensiolabs.com